Representation of Events in Nerve Nets and Finite Automata

Stephen C Kleene (1909-1994)

- Automata studies (1956)

Kleene's Regular Expressions

provides a language to program finite state automata, or to specify the behaviour of such machines.

Program execution is represented as a string of actions occurring sequentially while the program runs.

Operators and constants

- p;q is sequential composition
  - execution of p is followed by execution of q
- p **u** q is choice between p and q
  - only one of them is executed
- **1** is executed by doing nothing and making no change

The Laws of Regular Algebra

- p;(q;r) = (p;q);r — **;** associates — and so does **u**
- **1**;p = p = p;**1** — **1** is the unit of **;**

- p **u** q = q **u** p — **u** commutes
- p **u** p = p — **u** is idempotent

- p;(q **u** r) = p;q **u** p;r — **;** distributes thru **u** — and leftward as well

Refinement Ordering ≤ (below)

- p ≤ q means every execution of p is an execution of q,
  so p is more determinate – stronger
  and q is more abstract – weaker

- If p, is a program and r, s are specifications
  p ≤ r means p satisfies r
  r ≤ s means r implies s

- The algebra ignores the distinctions, because the proofs are the same for both specifications and programs

$p \leq q$ means $p \mathbin{\mathbf{u}} q = q$

Covariance

- Theorems:

  $p ; q \leq p ; (q \mathbin{\mathbf{u}} r)$
  $q ; p \leq (q \mathbin{\mathbf{u}} r) ; p$

- Proof rule:

$$\frac{p \leq q}{p ; r \leq q ; r \qquad r ; p \leq r ; q}$$

If the antecedents above the line have been proved, so are the consequences below the line.

More proof rules for ≤

$$\frac{p \le q \quad \& \quad q \le r}{p \le r}$$

(≤ is transitive)

$$\frac{p \le q \quad \& \quad q \le p}{p = q}$$

(≤ is antisymmetric)

All the rules can be derived from the laws, by a little proof

1. Proofs of programs

An Axiomatic Basis for Computer Programming

C.A.R. Hoare

Communications of the ACM 12(10) 1969

The Hoare triple

- Purpose: to prove that all possible executions of a program q when started after a precursor p will exhibit some desirable property r.

- Definition: {p} q {r} = p;q ≤ r

- Interpretation:
  - If p (the precondition) describes what has happened so far
  - and q is now started and executed to completion,
  - then the trace of overall execution will satisfy r (the postcondition).

Rule: Sequential composition

$$\frac{\{p\}\ q\ \{s\} \quad \& \quad \{s\}\ q'\ \{r\}}{\{p\}\ q;q'\ \{r\}}$$ (Hoare)

It is equivalent to a weaker form of the law of associativity

$p;(q;q') \leq (p;q);q'$ (Proof by covariance)

It is not possible from the rule to prove $(q';q);p \leq q';(q;p)$,

A Calculus of Communicating Systems

A.J.R.G Milner

Springer Lecture Notes in Computer Science 1980

Milner Transitions

- Purpose: to show how an implementation can generate just a single execution of the program r .
- Definition: $r \xrightarrow{q} p \quad = \quad q;p \leq r$
- Interpretation:
  - r may be executed by first executing q ,
  - with p as continuation for later execution.
  - (maybe there are other ways of executing r)

Rule: Sequential composition

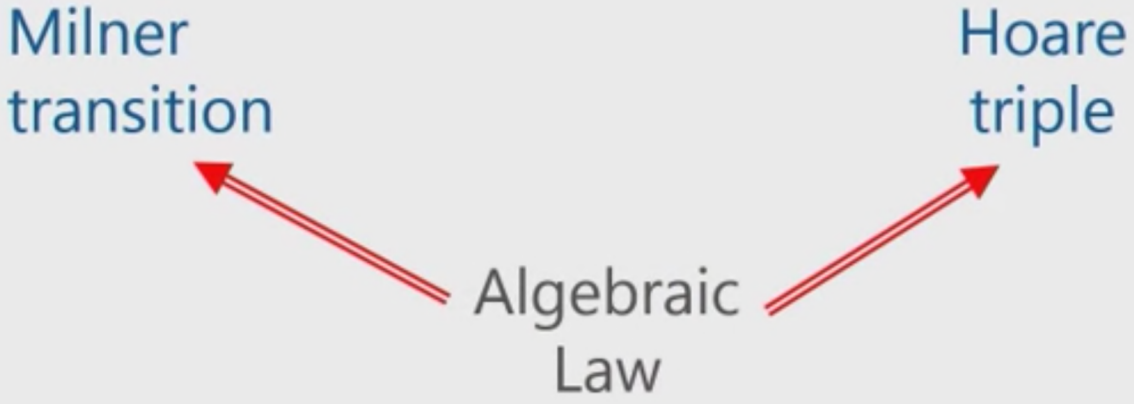
$$\frac{r \xrightarrow{q'} s \quad \& \quad s \xrightarrow{q} p}{r \xrightarrow{q';q} p}$$

is equivalent to the other weak form of associativity

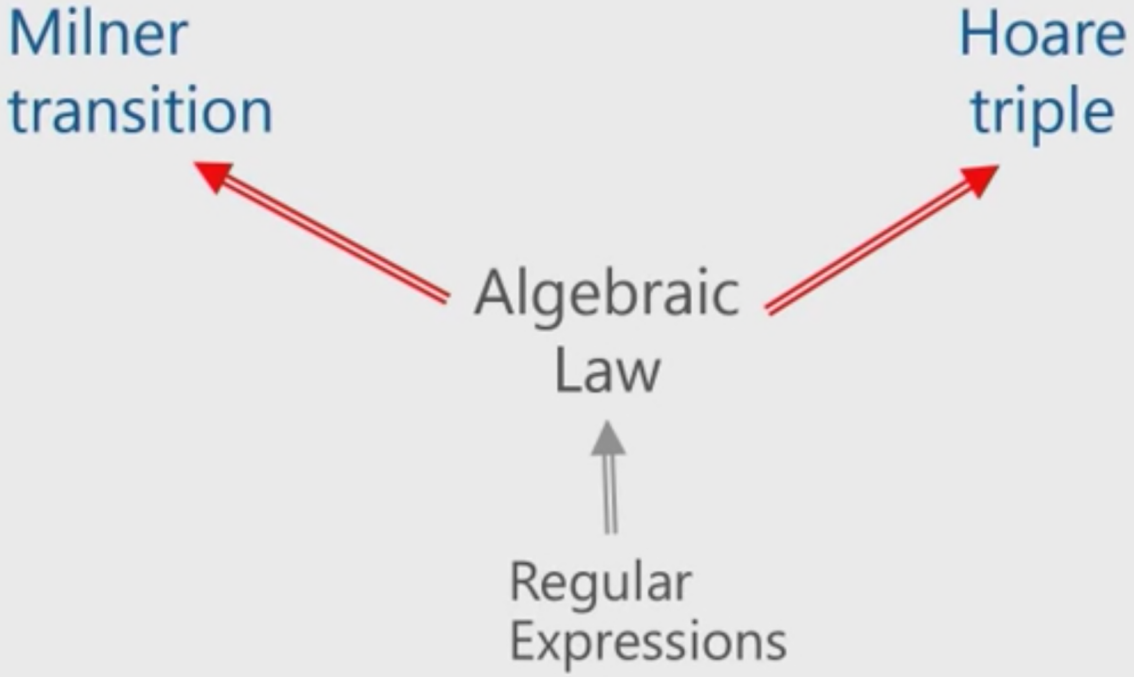
$$(q';q);p \leq q';(q;p),$$

By antisymmetry of $\leq$, conjunction of the Hoare inequality with the Milner inequality gives the associative equation.

Summary: Sequential Composition



Summary: Sequential Composition



Concurrent Composition: p||q

- p and q start together and finish together
  - in between, they may communicate.

- || is associative, **commutative**, and idempotent
- and distributes through **u**
- and has unit **1**

- and obeys the Exchange Law …

The exchange law

- Axiom: $(p \| q) ; (p' \| q') \leq (p ; p') \| (q ; q')$
- The two '$;$'s on the RHS of $\leq$ are scheduled to occur simultaneously, as shown by the single $;$ on the LHS.
- LHS is a more interleaving sequential implementation of the more general concurrency of RHS
- The LHS is a subset of the interleavings of the RHS

The Exchange Law

- Axiom: $(p \| q) \mathbin{;} (p' \| q') \leq (p \mathbin{;} p') \| (q \mathbin{;} q')$

- Theorems (frame laws):
  1. $(p \| q) \mathbin{;} q' \leq p \| (q \mathbin{;} q')$
  2. $p \mathbin{;} (p' \| q') \leq (p \mathbin{;} p') \| q'$
  3. $p \mathbin{;} q' \leq p \| q'$ and $q \mathbin{;} p' \leq p' \| q$

Proof: substitute **1** for the variable(s) of the axiom that are omitted in the theorem

Interleaving example

- Let a, b, c, d be atomic actions of a sequential program
  - written without semicolons as abcd
- Let x, y, z, w be atomic actions of another such program
  - written xyzw
- The two programs are executed concurrently
- the next slide shows how the Exchange Law permits an interleaved execution of actions from the two sequences

Interleaving by exchange

$$
\begin{array}{lcl}
 & & abcd \parallel xyzw \\
\text{(assoc ;)} & = & (a;bcd) \parallel (xy;zw) \\
\text{(exchange)} & \geq & (a \parallel xy) \,;\, (bcd \parallel zw) \\
 & = & (a \parallel x;y) \,;\, (b;cd \parallel zw) \\
\text{(frame)} & \geq & (a \parallel x) \,;\, y \,;\, (b \parallel zw) \,;\, cd \\
\text{(comm } \parallel \ldots) & \geq & xayzbwcd
\end{array}
$$

Different associations and commutations of ; and || at each step will obtain all other interleavings.

Proof Rules

for concurrent composition

Modular proof rule for ‖

$$\frac{p;q \leq r \qquad \& \qquad p';q' \leq r'}{(p \| p');(q \| q') \leq (r \| r')}$$

- Splits the proof of a complex inequality between concurrent programs into two simpler sequential proofs.
- The proof rule is proved from the exchange law and the exchange law is proved from the rule

Modularity rule implies the Exchange law

$$\frac{p;q \le r \quad \& \quad p';q' \le r'}{(p \| p');(q \| q') \le r \| r'} \quad \text{(modularity rule)}$$

- Replace r by $p;q$ and r' by $p';q'$
- The antecedents ($p;q \le p;q$ and $p';q' \le p';q'$) are now proved by reflexivity of $\le$
- and the conclusion is:

  $(p \| p');(q \| q') \le (p;q) \| (p';q')$

  which is the exchange law

Exchange law implies Modularity rule

- Assume the antecedents of the rule: $p;q \leq r$ and $p';q' \leq r'$

- $(p;q)\|(p';q') \leq (r\|r')$ (covariance of $\|$ 2ce)
- $(p\|p');(q\|q') \leq (p;q)\|(p';q')$ (exchange law)
- So $(p\|p');(q\|q') \leq (r\|r')$ (by transitivity of $\leq$)

- Therefore $\dfrac{p;q \leq r \quad \& \quad p';q' \leq r'}{(p\|p');(q\|q') \leq r\|r'}$ (the modularity rule)

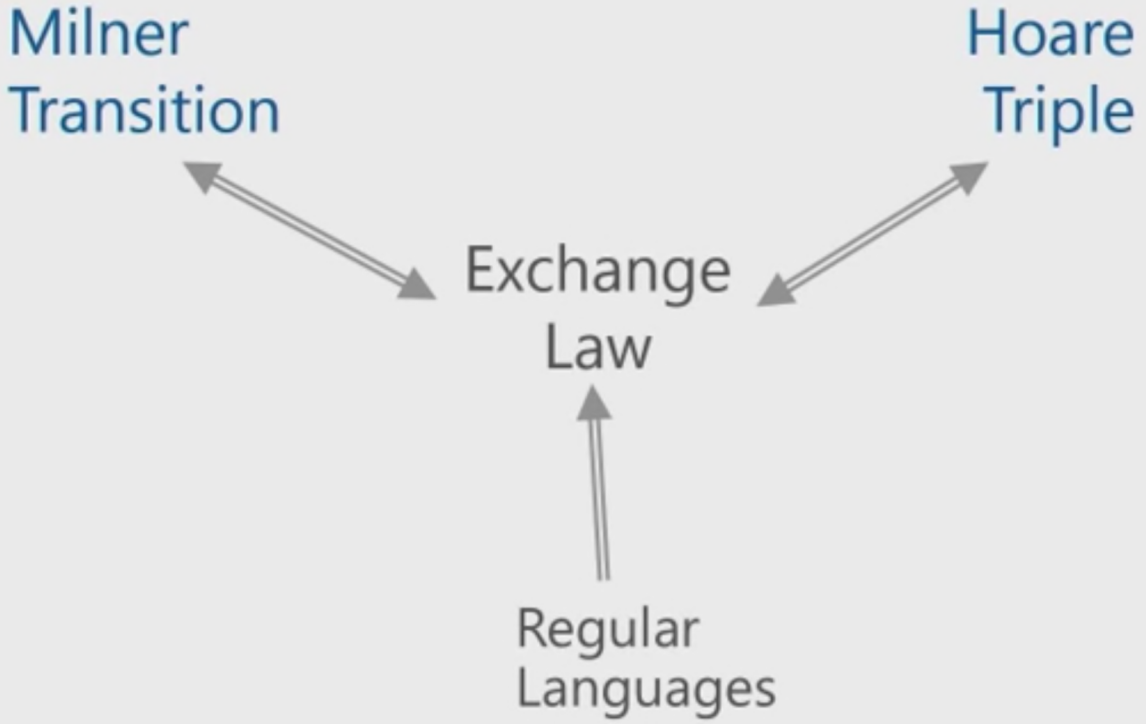
- $\dfrac{\{p\}\ q\ \{r\} \qquad \{p'\}\ q'\ \{r'\}}{\{p \| p'\}\ \ q \| q'\ \ \{r \| r'\}}$ Separation Logic (O'Hearn)

- $\dfrac{r \xrightarrow{q} p \qquad r' \xrightarrow{q'} p'}{(r \| r')\ \ q \| q' \xrightarrow{\ } (p \| p')}$ Operational Semantics (Milner's CCS)

- $\dfrac{p;q \leq r \qquad p';q' \leq r'}{(p \| p');(q \| q') \leq (r \| r')}$ modularity rule

are all equivalent to the Exchange axiom

Summary: Concurrent Composition



Microsoft

In Praise of Algebra

Algebraic Laws

- are taught to schoolchildren
- appreciated by mathematicians
- used by engineers
- and by software tools
  - optimisers, compilers,
  - program verifiers, generators and analysers,
  - test case generators, displays, and diagnostic aids
- They play a central role in unification of theories

Anybody against?